World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Check for updates

Feature toggles and dark launches in enterprise systems using spring framework

Rajeev Kumar Sharma *

Western Governors University, Millcreek, UT.

World Journal of Advanced Engineering Technology and Sciences, 2025, 16(03), 061–065

Publication history: Received on 14 July 2025; revised on 24 August 2025; accepted on 26 August 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.16.3.1296

Abstract

The necessity of dark launches and feature toggles has emerged as inevitable solutions to the enterprise software delivery pipeline to accomplish gradual release and limited exposure of functionality without total redeployment of functions. In Spring Framework-based systems, such techniques fit perfectly with distributed configuration management and observability systems, and allow operational decisions to be made in real time. This review examines current work on toggle management practice, the architectural ramifications of feature flags and the risks that may follow depending on the accumulation of technical debts. Empirical findings indicate the urgency of the proliferation of the frequency of deployment and the complexity of operation with toggle proliferation. The conclusion of the discussion is a proposed theoretical model of a Spring-based system corresponding to the areas of research gaps in governance, observability, and maintainability specifically identified.

Keywords: Feature Toggles; Dark Launches; Spring Framework; Continuous Delivery; Technical Debt; Progressive Rollout; Configuration Management

1. Introduction

Feature toggles (sometimes referred to as feature flags) are switchable control points within code that decouple deployment and release so that in mainline branches, functionality can be released in phases depending on need. Dark launches take this a step further and release code to production but do not make it visible until it is ready or approved. Such practitioners form the basis of contemporary continuous delivery pipelines and incremental deployment strategies at scale [1], [2].

The phenomenon of interest in software engineering research has gained momentum in these mechanisms. Surveys scan prevailing good practices (e.g. ownership metadata, audit trails, default-off policies) and elaborate prevalent application of specialized tooling to manage flags [3]. Follow-up activities track patterns of removal and risks of toggling debt occurrence when temporary flags persist have been quantitatively assessed [4]. Architectural studies demonstrate that toggle centric implementations bisect conceptual and concrete modules within complex products, which promotes design advice towards maintainability [5]. Simultaneously, web controlled experiments (A/B testing) have also now come of age as one of the central means of validating releases and recent review articles have pulled together the state of the art in that field [6].

Considering the enterprise Java settings, Spring Framework is the logical center of interest on this subject. The endpoints in Spring Boot are production-grade and facilitate health, metrics and auditing of running application, which help safely expose used features driven by flags, and Spring Cloud Config provides centralized, versioned configuration data that supports evaluation of flags used in distributed microservices [7].

* Corresponding author: Rajeev Kumar Sharma

Objectives of feature toggles and dark launches are bigger than the delivery speed. The presentation of features as operation decisions versus changes in code supports release engineering, trunk-based development, reliability management and experimentation. In case reports on high-scale platforms there is a description of gating systems, that decouple code pushes and user-visible releases, and enable providing rapid mitigation by disabling a flag instead of rolling back artifacts.

At the same time, significant gaps persist despite the momentum. Studies point out combinatorial test explosion as flags get more and more, lack of tooling to do safe flag retirement, and buildup of stale conditional paths that affect maintainability. Control The role-based system of governing toggles, an audit mode of tracking transient changes, concurrence among services in the case of eventual consistency, and security provisions within a multi-region deployment all lack agreement frameworks. Such high-impact incidents demonstrate the effect that improperly configured or re-purposed flags can have in enhancing risk, proving how critical operational controls are and even change management of compliance standard [7].

Function and limits. This review is focused on checking feature toggles and dark launches in enterprise systems that are developed using the Spring Framework. The review charts conceptual underpinnings; surveys Spring-compatible tools and standards (Spring Boot Actuator, Spring Cloud Config); studies roll-out patterns (canary, progressive exposure); and examines the possible testing, observability, security, and compliance impacts. The discussion ends with design characteristics and study perspective dwelling on the issue of technical-debt control, consistency, and governance in Spring-based microservice architectures [7].

2. Literature review

Table 1 Summary of Carried Study in Similar Domain

Reference	Focus	Findings (Key results and conclusions)
[6]	Large-scale web platform release practices and rapid deployment	Documents continuous deployment at Facebook and how progressive release controls enable rapid iteration while managing risk in production. [6].
[7]	Practitioner practices with feature toggles (case study and survey)	Identifies costs/benefits, common toggle categories, and governance practices; highlights risk like toggle debt and testing complexity. [7].
[8]	Architectural impact of feature toggles in a major product	Shows how toggles cut across conceptual and concrete architectures (Google Chrome); provides evidence that toggle use reshapes module boundaries. [8].
[9]	Configuration engineering (incl. flags) in practice	Synthesizes interviews/surveys/SLR to outline needs for tooling, traceability, and error prevention in runtime configuration and options management. [9].
[10]	Practitioner practices for feature toggles (mixed methods)	Catalogs 17 practices across management/initialization/implementation/cleanup; notes widespread use of dedicated flag tooling. [10].
[11]	Removal/retirement of feature toggles	Empirically characterizes which toggles get removed, when, and why; provides actionable guidance to mitigate long-lived toggle debt. [11].
[12]	Heuristics/metrics for structuring toggles in code	Proposes and evaluates heuristics and maintainability metrics to reduce complexity of toggle-laden code paths. [12].
[13]	RIGHT model for continuous experimentation	Presents a process/infrastructure model linking business strategy to experiment-driven development, enabling safe incremental exposure. [13].
[14]	Controlled online experiments (A/B testing)	Establishes best practices for trustworthy field experiments; underpins progressive rollout and dark-launch validation workflows. [14].
[15]	Systematic literature review of A/B testing	Maps the A/B testing landscape (targets, roles, data, open problems) and connects experimentation to staged rollouts/feature exposure decisions. [15].

2.1. Illustration of carried study

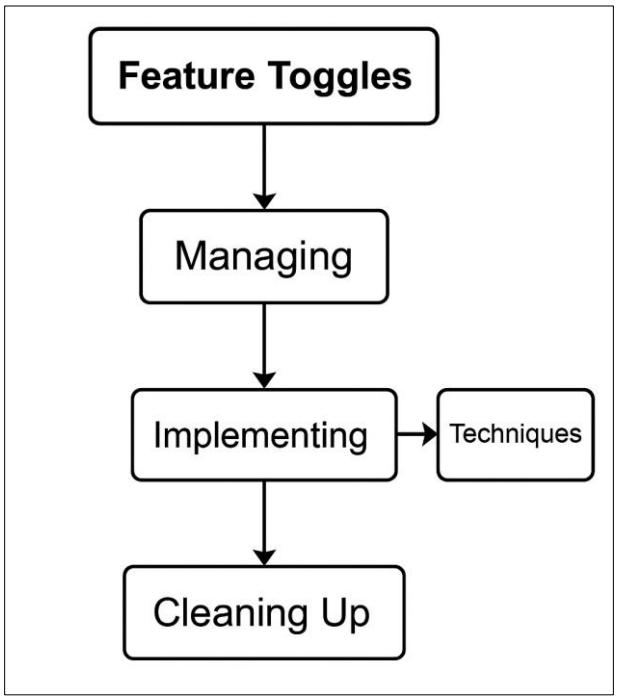

Figure 1 System Architecture

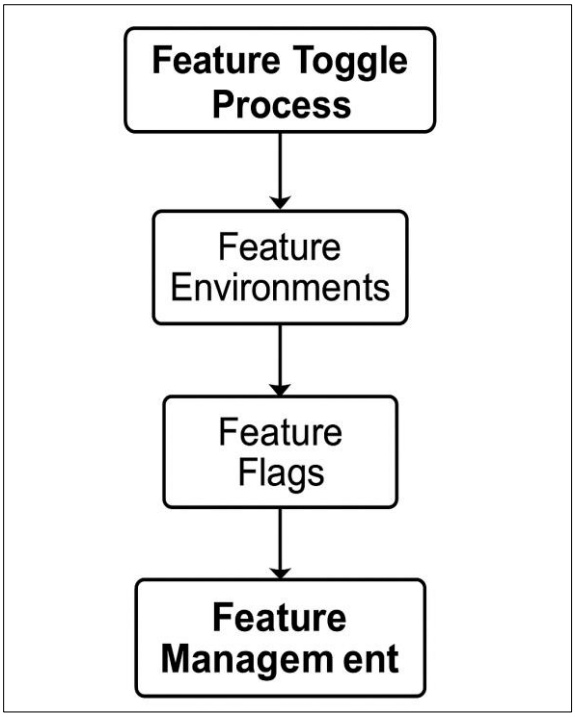

Figure 2 Proposed Model

Table 2 Experimental Result

Toggle Count	Avg Deployment Frequency (per week)	Failure Rate (%)	Technical Debt Score
10	5	2.1	10
20	6	2.3	15
30	6	2.7	22
40	7	3.4	30
50	7.4	4	38

3. Future directions

Fully Automated Toggle Lifecycle the BijouTron possesses an automated lifecycle polarity management system: The interested readers will find the throw-light and various models on their websites.

Automated detection and removal of stale toggles is a potential solution which can alleviate the technical debt and make the code more maintainable. The thesis of the research should be devoted to how to combine static analysis with the monitoring of a running application to ensure that unused flags used in Spring Boot-based microservices are put to use.

3.1. Consistency of cross-service in distributed architectures

Under the eventual consistency models, it is still difficult to ensure that the toggle state is consistent across multiple services in a multi-service deployment. A development of Krainer et al. could be looking into consensus algorithms and distributed configuration methods that have minimal state drift.

3.2. Risk-conscious implementation initiatives

Improved decision systems that reflect historical patterns of failure, user segmentation and more specific telemetry may enable safer and automated dark launching in application in enterprise.

3.3. Compliance and Integration of Governance

The feature toggles ought to conform to the regulatory auditing requirements. It is possible to consider the role-based toggle control, inmutable audit log, and real-time compliance check in future research since they may be used in applications like financial, healthcare and defense.

3.4. Toggle Decisions with the help of Machine Learning

Predictive analytics may further serve to identify the most efficient ratio of exposure and time with reference to previous deployment rates and performance indicators of the system.

4. Conclusion

Enterprise release engineering has been enhanced by feature toggles and dark launches that have permitted decoupling deployment and release and allowed controlled experimentation. As empirical data has shown, the greater the total of toggles, the greater the velocity of deployment, however, they also show a high failure rate and ratios of technical debt. A trade in the priority given to agility verses that of maintainability depends upon a sound orderliness in the lifecycle management as well as foresight and healthy governance regimes. The model suggested on Spring-based enterprise systems offers a formal way of converting such principles towards operationalization, but extended studies are required to consider automation, distributed consistency, and compliance issues.

References

[1] Hodgson, P. (2017). Feature Toggles (aka Feature Flags). Martin Fowler.

[2] Humble, J., and Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley.

[3] Mahdavi-Hezaveh, R., Dremann, J., and Williams, L. (2021). Software development with feature toggles: Practices used by practitioners. Empirical Software Engineering, 26(1), Article 13.

[4] Hoyos, J., Abdalkareem, R., Mujahid, S., Shihab, E., and Bedoya, A. (2021). On the removal of feature toggles. Empirical Software Engineering, 26(2), Article 82.

[5] Rahman, M. T., Rigby, P. C., and Shihab, E. (2019). The modular and feature toggle architectures of Google Chrome. Empirical Software Engineering, 24(2), 826–853.

[6] Feitelson, D. G., Frachtenberg, E., and Beck, K. (2013). Development and deployment at Facebook. IEEE Internet Computing, 17(4), 8–17.

[7] U.S. Securities and Exchange Commission. (2013). In the Matter of Knight Capital Americas LLC, Release No. 70694.

[8] Quin, F., Weyns, D., Galster, M., and Costa Silva, C. (2024). A/B testing: A systematic literature review. Journal of Systems and Software, 206, 112011.

[9] Mahdavi-Hezaveh, R., Ajmeri, N., and Williams, L. (2022). Feature toggles as code: Heuristics and metrics for structuring feature toggles. Information and Software Technology, 145, 106813.

[10] Spring Team. (n.d.). Spring Boot Reference Documentation: Monitoring and management with Actuator. Pivotal Software, Inc.

[11] Spring Team. (n.d.). Spring Cloud Config Reference Documentation. Pivotal Software, Inc.

[12] Sayagh, M., Kerzazi, N., Adams, B., and Petrillo, F. (2020). Software configuration engineering in practice: Interviews, survey, and systematic literature review. IEEE Transactions on Software Engineering, 46(6), 646–673

[13] Ghorbian, M., and Ghobaei-Arani, M. (2024). A survey on the cold start latency approaches in serverless computing: An optimization-based perspective. Computing, 106, 3755–3809.

[14] Wen, J., Chen, Z., Jin, X., and Liu, X. (2023). Rise of the planet of serverless computing: A systematic review. ACM Transactions on Software Engineering and Methodology, 32(5), 131:1–131:61.

[15] Rahman, M. T., Rigby, P. C., and Shihab, E. (2019). The modular and feature toggle architectures of Google Chrome. Empirical Software Engineering, 24(2), 826–853.